Evaluation Scope for Ubiquitous Computing

Larry Arnstein*, Jong Hee Kang*, Gaetano Borriello**
*Department of Computer Science & Engineering, University of Washington
**Intel Resarch Labs, Seattle, WA

Ubiquitous computing presents a challenging evaluation problem because we must do without many of the standard assumptions allowed in pure HCI evaluations. As an example, the user's experience does not start once an application is successfully loaded and running in the chosen client device with all required resources. Instead, it starts when a hardware or software artifact is conceived by a developer; and it continues through the development, adoption, installation, use, and retirement of that artifact. As a result, the people in our scenarios are not limited to just the end-users of the system. Instead, we must be concerned with the experience of developers, system administrators, and even people who may want to abuse the system. Our position is that the ubiquitous computing community can draw on lessons learned from the HCI community in terms of its focus on the user experience but with broader scope, and from the from the systems community in terms of its quantitative understanding of system properties but with more focus on the subjective experience of specific stakeholders. Matching up stakeholders with system properties should help in the design appropriate evaluation experiments.

Our work on the problem of experiment capture in a cell biology laboratory has raised some of these issues. Labscape [1], as our smart space is called, includes tagging technology to track objects and people, wired and wireless networked instrumentation, wireless audio I/O, and ubiquitous flat panel displays. Our primary goal is for the biologists to perceive the capture system as a flexible lab assistant that presents relevant information to the experimenter based on the specific task at hand. While it is necessary to evaluate this experience using standard HCI techniques, it is not sufficient because successful deployment depends on how the properties of the entire system are perceived by developers and system administrators.

The current state of the Labscape system provides some examples of the issues that arise when the scope of evaluation goes beyond the domain of HCI. Our smart laboratory system includes the following software components:

- The Location Server that keeps track of the location of people and objects with respect to named locations.
- Resource managers that moderate access to the resources of the named locations requested by migrating applications.
- Virtualized devices drivers that provide local I/O to visiting applications. Virtualization is necessary because the applications do not have to actually migrate to the local processor to access these resources.
- The lab-assistant/capture application that follows the user around, requesting local resources from local resource managers. Examples of such resources are virtualized device drivers and local displays

These components were built to provide the minimum distributed system functionality required for evaluation of the end-user's experience in a laboratory environment under typical assumptions found in the HCI community -- *that the application is always available and running successfully on the closest client in the lab.* Though entirely adequate for that purpose, the current system would fail a broader ubiquitous computing assessment on the criteria listed below. Our point is not to address the virtues or weaknesses of our particular system or how they could be overcome, but rather to highlight some system characteristics that should be assessed in a ubiquitous computing application and the stakeholders that would be involved. We are in the process of designing evaluations experiments to cover all of these issues and look forward to the outcome of the workshop.

Configurability and the System Administrator. Each application must be configured with the name of the location server, and the names and port addresses of the various virtualized resources required by the application. This information must match the configuration of the resources managers and virtual resource services running on behalf of each named location. Other points of configuration include programming the user's active badge with the user's ID, assigning devices to named locations, and setting up all of the machines in the system so that they have a consistent view of the network for access to centralized code and data. This is primarily an issue for the system administrator who would be responsible for setting up the environment. One goal of ubiquitous computing should be to minimize the need for system administration.

Robustness and the User. Due to the large number of network services cooperating over socket connections, the robustness of our system was quite low. Though applications could be stopped and started without restarting the location server (if they were shut down property), it was not possible to create new named locations without restarting. Furthermore, it was not possible to restart the location server without restarting all of the resource managers and applications. The protocols between components were not designed with failure handling in mind. For example, the resource manager locks resources until they are released by the application. Deadlock results if the application dies without releasing. The lack of robustness would primarily impact the user, as the system would often not be available when needed.

Extensibility and the Developer. In this category, we include efforts required to add new functionality to the system. The requirements of our system may place a special burden on the hardware and software developers. If this burden were too high or specialized, then the developer community would hesitate to support it. A hardware example is that if I/O devices are able to attach some device information to the data, system configuration can be simplified. From a software perspective application programmers in our system must instantiate a special mobility object that knows how to interact with the location server and resource managers. The application programmer is simply required to register resource type requirements with the mobility object and to provide for serializability of the application. The developer and possibly the system administrators would the major stakeholders in assessing extensibility.

Some of these examples lend themselves best to a checklist or heuristic style of evaluation. The evaluator would test whether or not certain properties exist, such as the ability to power cycle any device in the system without loss of data or functionality. Others would require controlled experiments for quantification. For example, one could measure the ability of minimally trained system administrators or end-users to add new hardware or software to an existing system or to move devices within the environment. The subjective and objective experiences could be assessed using known HCI techniques.

This is not intended to be an exhaustive list of system properties or stakeholders that are relevant to the ubiquitous computing community, rather it is intended to make the following point: *evaluation of ubiquitous computing systems must cover the entire lifespan of the system. From design, development, deployment, and maintenance, and all of its stakeholders including developers, administrators, and end-users.* The question that I hope to explore in the workshop is: which methodologies and techniques are most appropriate for assessing the experience of each and how should these results be quantified?

[1] Arnstein, L. F., Sigurdsson, S.,, Franza, R., "Ubiquitous Computing in the Biology Laboratory", *Journal of Lab Automation (JALA).* vol 6, no. 1, March 2001. http://labscape.cs.washington.edu